High Resolution Finite Volume Parallel Simulations of Mould Filling and Binary Alloy Solidification on Unstructured 3-D Meshes*

A.V. Reddy†, D.B. Kothe¶, C.Beckermann§, R.C.Ferrell+, and K.L. Lam#

Abstract

The Los Alamos National Laboratory (LANL) is currently developing a new casting simulation tool (known as Telluride[1]) that employs robust, high-resolution finite volume algorithms for incompressible fluid flow, volume tracking of interfaces, and solidification physics on 3-D unstructured meshes. The finite volume algorithms are efficient, parallel, and robust, and are based on colocated cell-centered schemes that are formally second order in time and space. The flow algorithm is a 3-D extension of recent efforts on projection method solutions of the Navier-Stokes (NS) equations. Our piecewise-planar volume tracking algorithm can accurately track the topologically-complex free surfaces present during mold filling. Coupled to our flow algorithm is a comprehensive binary alloy solidification model that incorporates macroscopic descriptions of heat transfer, solute redistribution, and melt convection, as well as a microscopic description of segregation. Our algorithms yield high-fidelity solutions on a variety of meshes, ranging from structured/orthogonal meshes to fully-unstructured (finite element) meshes. Key computer science developments have enabled us to efficiently parallelize our unstructured mesh algorithms on both distributed and shared memory computing platforms. These include our object-based use of Fortran 90 and new parallel libraries for gather/scatter functions (PGSLib) and solutions of linear systems of equations (JTpack90). Examples of our current capabilities are illustrated with simulations of mold filling and solidification of 3-D components currently being poured in LANL foundries.

Introduction

Driven by demands on quality and control of microstructure of materials, modeling of casting processes is being increasingly relied upon to predict potential solidification defects and thus improve casting practices, reduce foundry costs, and aid in the design of new and improved materials. Casting simulation tools must capture complicated cast geometries, necessitating the use of unstructured meshes. Most finite volume algorithms, however, are restricted to orthogonal meshes, and hence stair-step boundaries, thus introducing unnecessary errors. Simulation accuracy also depends on the number of computational cells used to partition the physical domain. Unless a simulation can be performed in parallel (where multiple processors are coordinated to perform the work simultaneously), CPU time and memory requirements become unacceptable when desired simulation accuracy and length scale resolution calls for large meshes. This is especially true in 3-D, hence parallelism is imperative in a modern casting simulation tool.

Physical Model

We are currently modeling alloy solidification with a simplified version of the volume averaged two-phase model of Ni and Beckermann [1], where the solid phase is assumed to be stationary. The governing equations are summarized in reference 2, and include mass, momentum, energy and species conservation equations that are valid in the single-phase liquid and solid regions as well as in the mushy zone. They are supplemented by phase diagram relations, a permeability expression, a back-diffusion model, and others. Simplified energy and species equations are presented in Table 1 to facilitate discussion of the phase change algorithm below.

*Supported by the Department of Energy Accelerated Strategic Computing Initiative Program.
†LANL, Fluid Dynamics Group T-3, Theoretical Division, Los Alamos, NM 87545 (*anand@lanl.gov*)
¶LANL, Fluid Dynamics Group T-3, Theoretical Division, Los Alamos, NM 87545 (*dbk@lanl.gov*)
§Dept. of Mech. Eng., Univ. of Iowa, Iowa City, IA 52245 (*becker@icaen.uiowa.edu*)
+Cambridge Power Computing Associates Ltd., 2 Still St., Brookline, MA 02146 (*ferrell@cpca.com*)
#LANL, Engineering Analysis Group ESA-EA, Los Alamos, NM 87545 (*klam@lanl.gov*)
[1] For more information on Telluride, see http://gnarly.lanl.gov/Telluride/Telluride.html.

Numerical Method

The system of conservation equations are integrated over a control (cell) volume to yield their integral (weak) form. Volume integrals are then converted to face integrals, which are approximated as discrete sums over faces. Second-order spatial and temporal accuracy results when face quantities are estimated with monotonicity-preserving time-centered Taylor series expansions. Our overall solution method integrates one time step with a series of fractional substeps, where each substep approximates a selected term of the conservation equations (e.g., advection, viscosity, etc.). This fractional-step method allows our algorithms to be developed in an accurate and modular fashion, and enables easy addition of new physical models.

Incompressible Flow Algorithm

LANL has a long history and made important contributions in the development of computational methods for modeling incompressible interfacial (e.g., free surface) flows. A particularly important contribution occurred in 1965 with the Marker-and-Cell (MAC) method, which remains (along with its successors) a popular choice to this day. A recent example of a MAC successor is RIPPLE [3], which has become a very popular model for 2-D free surface flows because of its algorithmic and physical model improvements, most notably the continuum surface force (CSF) model for surface tension [4]. Since the inception of the MAC method, numerous advances have taken place in the development of incompressible interfacial flow algorithms. Perhaps most notable is the recent work of Bell, Colella, and coworkers [5,6] in devising high resolution projection method solutions of the Navier-Stokes (NS) equations coupled with modern interface tracking algorithms [7]. This approach has yielded high-fidelity flow solutions that are fully second-order in time and space. The methodology borrows in large part from algorithms devised for high-speed flow, by coupling projection methods with high-order Godunov advection and interface tracking.

Analysis and refinement of these projection methods has resulted in an overall NS solution algorithm that is robust and accurate [8]. We have incorporated material interface variable-density effects (e.g., due to free surfaces) into our flow algorithm with the addition of various modern interface tracking methods [9]. Because of our need to model LANL gravity-pour mold-filling scenarios, we have considered only those interface tracking algorithms capable of following surfaces that undergo gross topology change. In designing interface tracking methods, they must extend to 3-D and have no restrictions on the topological complexity or the number of interfaces that may be represented. After extensively studying many possible approaches [9], our design constraints have led us to PLIC (piecewise-linear interface calculation) volume tracking methods [10,11].

We have extended our PLIC algorithm to generalized 3-D hexahedra grids [7]. Our algorithm can track topologically complex fluid interfaces on these meshes, while maintaining an interfacial width of one cell. This is in contrast to high-order continuum advection schemes, where interfacial discontinuities can be smeared over several cells after advection and/or topological change [9]. We estimate interface geometry from local fluid volume data, and assume interfaces to be locally planar within each cell. Knowledge of the interface geometry allows partitioning of total volume fluxes into individual material volume fluxes. Total fluid volume, by construction, is conserved rigorously.

We have extended our projection-based NS solution method to 3-D unstructured grids without needlessly sacrificing robustness, accuracy, or efficiency. Our current approach has borrowed in part from techniques originating in the aerodynamics community, an example being the least-squares reconstruction schemes devised by Barth [12], which maintain high fidelity on complex unstructured meshes. We have also extended a 3-D unsplit advection technique [13] to unstructured meshes, which avoids operator-splitting and allows consistent use of high-order monotone advection in incompressible flows.

Phase Change Algorithm

Phase change in pure materials occurs at a constant temperature. In such situations the phase change rate is obtained from solutions to an enthalpy conservation equation. However, alloy solidification is more complicated, requiring solutions to both enthalpy and species conservation equations. The species and enthalpy conservation equations given in Table 1 are intimately coupled through the phase change rate term ($\partial(\rho_s\varepsilon_s)/\partial t$). To achieve efficient convergence in the iterative solution of this system of equations, a good estimate for the phase change rate is required. Several methods have been proposed for updating the phase change rate during each solution iteration. Schneider and Beckermann [2] propose a method which explicitly solves for the phase change rate by combining the discretized enthalpy and species equations. Voller and Swaminathan [14] propose methods in which the phase change rate is linearized as a truncated Taylor series, and old iteration values are then used to estimate the linear term. Here we use a different approach, in which the phase change rate is estimated using an expression derived from the binary phase diagram and the species conservation equations. This method is efficient and robust.

In our iterative scheme, we estimate the energy source due to phase change (Table 1) at iteration level m+1 as

$$S = (h_l - h_s)\partial(\rho_s\varepsilon_s)/\partial t \cong (h_l - h_s)(\rho_s^{m+1}\varepsilon_s^{m+1} - \rho_s^o\varepsilon_s^o)/\delta t\,, \tag{1}$$

where the superscript o refers to the previous time step. Following Voller and Swaminathan [14], a truncated Taylor series expansion for $\rho_s^{m+1}\varepsilon_s^{m+1}$ gives

$$\rho_s^{m+1}\varepsilon_s^{m+1} = \rho_s^m\varepsilon_s^m + \frac{d(\rho_s\varepsilon_s)}{dT}(T^{m+1} - T^m), \tag{2}$$

which, when substituted into Eq. (1), yields a linear source term (in temperature) in the enthalpy equation. A a good estimate for $d(\rho_s\varepsilon_s)/dT$, however, is needed before efficient convergence can be realized. An outline of the derivation of a general expression for this term follows.

After discretizing and rearranging the solid species equation (Table 1), one can obtain

$$\rho_s\varepsilon_s C_s = \rho_s^o\varepsilon_s^o C_s^o + C_{si}\delta(\rho_s\varepsilon_s) + \frac{S_v\rho_s D_s}{l_s}\delta t(C_{si} - C_s), \tag{3}$$

and, a similar discretization of the liquid species equation yields

$$\rho_l^o\varepsilon_l^o C_l = \rho_l^o\varepsilon_l^o C_l^{'} + (C_l - C_{si})\delta(\rho_s\varepsilon_s) + \frac{S_v\rho_s D_s}{l_s}\delta t(C_s - C_{si}), \tag{4}$$

where $C_l^{'}$ is the liquid species concentration after advective transport, i.e., $C_l^{'} - C_l^o$ is the species concentration change due to advection (flow). (Note that solid is stationary and hence its concentration does not change during the advection step).

We can derive an expression for $d(\rho_s\varepsilon_s)/dT$ by performing the following three steps. First, substitute binary phase diagram relations (for liquid and interfacial solid concentrations) in Eqs. (3) and (4). Second, differentiate the resulting equations with respect to temperature. Finally, eliminate the term for the rate of change of solid concentration with temperature from the equations resulting from step 2, thus obtaining the required expression. Details of this derivation will be in a forthcoming paper.

If the phase change is isothermal, a few comments about our numerical scheme are in order. Voller and Swaminathan[14] set $d(\rho_s\varepsilon_s)/dT$ to a large value (large source term procedure) to fix the temperature of the control volume undergoing solidification at the melting point value. Our scheme employs Krylov-subspace methods, and a large value for $d(\rho_s\varepsilon_s)/dT$ unnecessarily increases the matrix condition number. This restricts the ability of these methods to converge to the desired tolerance. To overcome this difficulty, the neighboring cell coefficients are set to zero except the reference (diagonal) coefficient, which is set to unity. The RHS of the matrix equation is set to the melting point temperature. This procedure, however, yields a non-symmetric matrix, hence GMRES is employed to solve the set of equations. This procedure increases the robustness of our algorithm with little additional computational effort.

Parallelization Strategy

Given our use of implicit finite volume algorithms that frequently invoke indirect addressing (because of the unstructured mesh connectivities), surprisingly high parallel efficiency (> 85%) has been regularly attained for our casting simulations [15-17]. We summarize briefly our parallelization approach here, and refer the interested reader to reference 15 for further details.

Our strategy for parallelization is to explicitly decompose and distribute the global mesh across all processors available to perform work on the problem at hand. This strategy is independent of the whether the memory accessible to the processors is local (distributed memory systems) or global (shared memory systems). We do not choose to rely upon compiler directives (as in High Performance Fortran) or compiler "switches" for parallelism. We have instead explicitly designed the parallelism into our software by separating all communication (indirect addressing) from computation. All indirect addressing functions are performed in gather/scatter procedures that are themselves parallelized by the explicit passing of messages between processors via calls to the MPI library. Platform-specific, explicit parallelism appears only in these procedures (instead of being littered throughout the entire source), which in turn call upon PGSLib [16] to actually perform the MPI-based message passing.

A given time step can require several matrix solutions, so the majority of our solution algorithm is spent in the JTpack90 linear solver library [17]. We currently solve our systems with JTpack 90 in parallel over the entire mesh, rather than invoking a Schwarz decomposition [18]. For orthogonal meshes, we store the matrix and use preconditioned CG to solve the system. For generally nonorthogonal, unstructured meshes, we do not store the matrix and use preconditioned GMRES to solve the system. In all cases, we interface with JTpack90 in matrix-free form, i.e., matrix-vector multiplication is performed with procedures provided by Telluride. This approach avoids having to assimilate and store the matrix, which for a generally unstructured mesh can be intractable.

Results

We now present filling and solidification simulations that are representative of our current capabilites. Additional simulation results (including animations) can be found elsewhere[1]. Since fluid flow/heat transfer coupling has not yet been implemented, the simulations presented here assume filling and solidification to be separate processes, i.e., an isothermal fill is simulated, after which we simulate solidification of the quiescent melt.

Solidification of an Al-4%Cu Ingot

An Al-4%Cu alloy is solidified in a 0.05x0.1x0.1m box by extracting heat from four points on the x=0 plane by maintaining these points at 600K. Initially the melt is assumed to be superheated by 5K at 921K. (The Al-Cu eutectic temperature is 821K.) A mesh of 16x32x32 (16384) cubic orthogonal cells is employed to discretize the domain.

Figure 1 shows isosurfaces of four different temperatures 195 seconds after the initial extraction of heat. The isosurfaces corresponding to 821K and 916K are the eutectic and liquidus isosurfaces, respectively, hence they demarcate the mushy zone. The isosurfaces close to the heat extraction points are hemispherical, as expected.

Filling and Solidification of a Copper Chalice

A more realistic problem, solidification of a copper "chalice", is chosen for the second simulation presented here. The copper chalice (cast at a LANL foundry) is essentially a hemispherical shell gated at the pole with a cylindrical riser, which serves to continuously supply liquid copper to the shell during solidification (to avoid shrinkage defects).

One quadrant of the chalice is simulated, with the geometric model and computational mesh (6480 unstructured hex elements) being generated with the I-DEAS commercial software package. A filling simulation is performed to model the melt being gravity-poured into the riser. The total filling time is approximately two seconds. Figure 2 shows the fill at 0.5 seconds in one half of the chalice (one quadrant result is reflected).

For the solidification simulation, approximate boundary conditions were estimated from the experimental data. The initial temperature of the melt is 1543K. A convective heat transfer coefficient of $25 W/m^2$-K is used for the inner surface of the hemisphere and the bottom of the chalice. For the outer surface a value of 15W/m^2-K is used. The top surface is assumed insulated because of its proximity to a heater. Figures 3 shows the liquid volume fraction field 425 seconds after initial extraction of heat. It can be noticed from Fig.3 that liquid in the riser solidifies last and thus no shrinkage defects can be expected in the shell. The total solidification time is about 10 minutes.

A higher-resolution (46386 cells) simulation of this problem was performed on eight processors of a 300MHz Digital AlphaServer 8400. Using the Chaco [19] decomposition software, the mesh was decomposed into eight submeshes (see Figure 4). An impressive parallel efficiency (an approximate speedup of seven) was achieved. Additional parallel efficiency results can be found in references 15 and 16.

Nomenclature

ε volume fraction	ρ density(kg/m^3)	C concentration(wt pct)
D mass diffusivity(m^2/s)	h enthalpy(J/kg)	k conductivity(W/m/K)
l back diffusion length (m)	S_v interfacial area concentration(m^{-1})	
T temperature(K)	t time(s)	v velocity(m/s)
Subscripts:		
i interfacial/summation variable	l liquid	s solid
Superscripts:		
m, m+1 iteration level	o old time step variable	' after advection step

References

1. J.Ni and C.Beckermann, Metall Trans.B, 1991, **22B**, 349.
2. M.C.Schneider and C.Beckermann, Metall Trans.A, 1995, **26A**, 2373.
3. D.B. Kothe and R.C. Mjolsness, AIAA Journal, 1992, **30**, 2694.
4. J.U. Brackbill, D.B. Kothe, and C. Zemach, Journal of Computational Physics, 1992, **100**, 335.
5. J.B. Bell, P. Colella, and H.M. Glaz, Journal of Computational Physics, 1989, **85**, 257.
6. J.B. Bell and D.L. Marcus, Journal of Computational Physics, 1992, **101**, 334.
7. E.G. Puckett, A.S. Almgren, J.B. Bell, D.L. Marcus, and W.J. Rider, Lawrence Livermore National Laboratory Report UCRL-JC-120451, to be published in the Journal of Computational Physics, 1997.
8. W.J. Rider, D.B. Kothe, S.J. Mosso, J.H. Cerutti, and J.I. Hochstein, Technical Report AIAA 95-0699, presented at the 33rd Aerospace Sciences Meeting and Exhibit, Reno, NV, 1995.
9. W.J. Rider and D.B. Kothe, Technical Report AIAA 95-1717, presented at the 12th AIAA CFD Conference, San Diego, CA, 1995.
10. D.B. Kothe and W.J. Rider, Los Alamos National Laboratory Report LA-UR-94-3384, 1995.
11. D.B. Kothe, W.J. Rider, S.J. Mosso, J.S. Brock, and J.I. Hochstein, Technical Report AIAA-96-0859, presented at the 34th Aerospace Sciences Meeting and Exhibit, Reno, NV, 1996.
12. T.J. Barth, Technical Report AIAA-89-0366, presented at the 27th Aerospace Sciences Meeting and Exhibit, Reno, NV, 1989.
13. J. Saltzman, Journal of Computational Physics,1994, **115**, 153.
14. V.R.Voller and C.R.Swaminathan, Numerical Heat Transfer, Part B, 1991, **19**, 175.
15. D.B.Kothe, R.C. Ferrell, J.A. Turner, and S.J. Mosso, in Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, 1997.
16. R.C. Ferrell, D.B. Kothe, and J.A. Turner, in Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, 1997.
17. J.A. Turner, R.C. Ferrell, and D.B. Kothe, in Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, 1997.
18. T.J. Barth, Technical Report AGARD Publication R-807, lecture notes presented at the VKI/NASA/AGARD Lectures Series on Parallel Computing, 1995.
19. B.Hendricson and R.Leyland, The Chaco User Guide: version 2, technical report SAND94-2692, Sandia National Laboratories, Albuquerque, NM, 1995.

Table:1 - Energy and Species Equations

Mixture energy: $$\left(\Sigma\rho_i\varepsilon_i\frac{\partial h_i}{\partial T}\right)\frac{\partial T}{\partial t}+\varepsilon_l\rho_l\frac{\partial h_l}{\partial T}(v_l\cdot\nabla T) = \nabla\bullet[(\Sigma\varepsilon_i k_i)\nabla T]+(h_l-h_s)\frac{\partial}{\partial t}(\rho_s\varepsilon_s)$$

Liquid species: $$\varepsilon_l\rho_l\frac{\partial C_l}{\partial t}+\varepsilon_l\rho_l v_l\cdot\nabla C_l = (C_l-C_{si})\frac{\partial}{\partial t}(\rho_s\varepsilon_s)+\frac{S_v\rho_s D_s}{l_s}(C_s-C_{si})$$

Solid species: $$\varepsilon_s\rho_s\frac{\partial C_s}{\partial t} = (C_{si}-C_s)\frac{\partial}{\partial t}(\rho_s\varepsilon_s)+\frac{S_v\rho_s D_s}{l_s}(C_{si}-C_s)$$

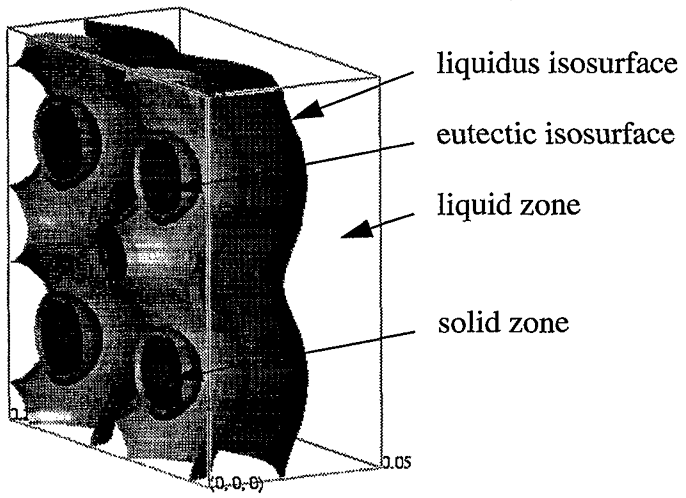

Fig. 1 Temperature isosurfaces during Al-Cu box solidification (195s)

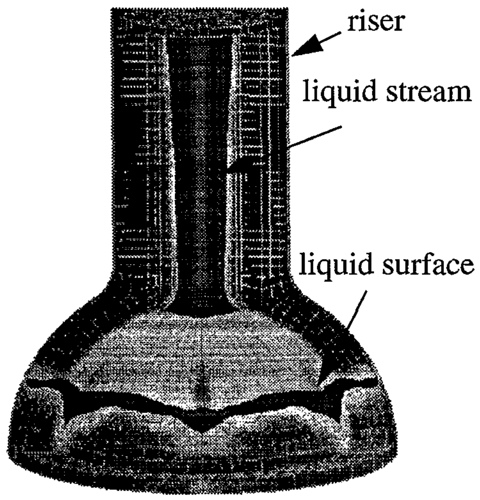

Fig. 2 Filling of a copper chalice (0.5s)

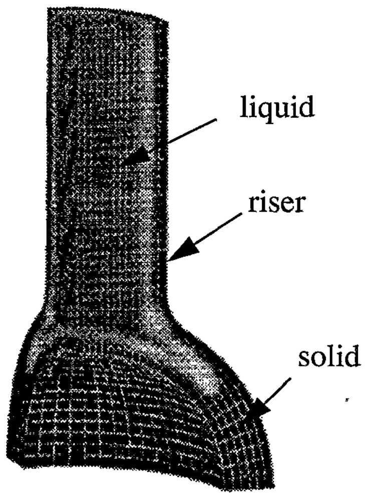

Fig. 3 Solid/liquid zones during copper chalice solidification (425s)

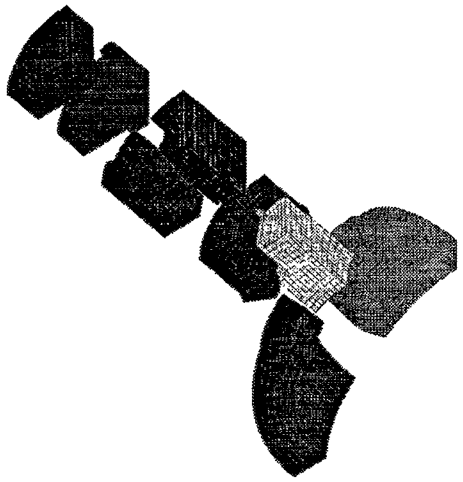

Fig. 4 Exploded view of decomposed mesh for 8 processors